Синтаксис

; (точка с запятой)

Синтаксис **; (точка с запятой)** используется для обозначения конца оператора.

Пример

```
int a = 13;
```

Подсказка

Забытая в конце строки точка с запятой приводит к ошибке компиляции. Текст ошибки может быть либо видимым и ссылаться на пропущенную точку с запятой, либо нет. Если встречается непонятная или похожая на нелогичную ошибка компиляции, одним из первых действий должна быть проверка пропущенных точек с запятой, в коде, непосредственно предшествующем строке, в которой компилятор выдал предупреждение.

Комментарии

Комментарии – это строки в программе, которые используются для информирования вас самих или других о том, как работает программа. Они игнорируются компилятором и не экспортируются в процессор

Есть два способа пометить строку как комментарий:

```
X = 5; // Это комментарий одной строки
/*
If (d==0){      Это многострочный комментарий
}
*/
```

Подсказка

Во время экспериментов с кодом, «закомментирование» частей программы – подходящий способ удаления строк, в которых могут быть ошибки. Так строки в коде остаются, но превращаются в комментарии, и компилятор просто игнорирует их. Это может быть особенно полезно при локализации проблемы, или когда не получается скомпилировать программу, а сообщение об ошибке при компиляции скрыто или бесполезно.

Фигурные скобки **{}** (также называются просто «скобки») – важный элемент языка программирования С.

Открывающая скобка “**{**” должна всегда сопровождаться закрывающей скобкой “**}**”. Это условие, известное как парность (симметричность) фигурных скобок.

Поскольку использование фигурных скобок столь многогранно, хорошей практикой программирования будет печатать закрывающую фигурную скобку сразу после того, как напечатана открывающая скобка, когда вставляется конструкция, для которой нужно использовать фигурные скобки. Затем возвращаем курсор в позицию между фигурными скобками и начинаем вводить операторы.

Функции

```
1  void НазваниеФункции(тип данных аргумента){
2
3      оператор(ы)
4
5  }
```

Циклы

```
while (логическое выражение)
 {

     оператор(ы)

 }
```

```
do
{

   оператор(ы)

} while (логическое выражение);
```

```
for (инициализация; условие окончания цикла; приращения цикла)
{

   оператор(ы)

}
```

Условные операторы

```
01 if (логическое выражение)
02 {
03
04     оператор(ы)
05
06 }
07 else if (логическое выражение)
08 {
09
10     оператор(ы)
11
12 }
13 else
14 {
15
16     оператор(ы)
17
18 }
```